Data multiplexing in radio interferometric calibration

Yatawatta, Sarod; Diblen, Faruk; Spreeuw, Hanno; Koopmans, L.V.E.

Published in:
Monthly Notices of the Royal Astronomical Society

DOI:
10.1093/mnras/stx3130

Publication date:
2018

Monthly Notices
of the
ROYAL ASTRONOMICAL SOCIETY

MNRAS **475,** 708–715 (2018)
Advance Access publication 2017 December 5

doi:10.1093/mnras/stx3130

Data multiplexing in radio interferometric calibration

Sarod Yatawatta,[1]★ Faruk Diblen,[2] Hanno Spreeuw[2] and L. V. E. Koopmans[3]

[1]*ASTRON, Postbus 2, NL-7990 AA Dwingeloo, the Netherlands*
[2]*Netherlands eScience Center, Science Park 140, NL-1098 XG Amsterdam, the Netherlands*
[3]*Kapteyn Astronomical Institute, University of Groningen, PO Box 800, NL-9700 AV Groningen, the Netherlands*

Accepted 2017 November 28. Received 2017 November 28; in original form 2017 July 6

ABSTRACT

New and upcoming radio interferometers will produce unprecedented amount of data that demand extremely powerful computers for processing. This is a limiting factor due to the large computational power and energy costs involved. Such limitations restrict several key data processing steps in radio interferometry. One such step is calibration where systematic errors in the data are determined and corrected. Accurate calibration is an essential component in reaching many scientific goals in radio astronomy and the use of consensus optimization that exploits the continuity of systematic errors across frequency significantly improves calibration accuracy. In order to reach full consensus, data at all frequencies need to be calibrated simultaneously. In the SKA regime, this can become intractable if the available compute agents do not have the resources to process data from all frequency channels simultaneously. In this paper, we propose a multiplexing scheme that is based on the alternating direction method of multipliers with cyclic updates. With this scheme, it is possible to simultaneously calibrate the full data set using far fewer compute agents than the number of frequencies at which data are available. We give simulation results to show the feasibility of the proposed multiplexing scheme in simultaneously calibrating a full data set when a limited number of compute agents are available.

Key words: instrumentation: interferometers – Methods: numerical – Techniques: interferometric.

1 INTRODUCTION

In order to reach many scientific goals of modern radio astronomy, large amounts of interferometric data need to be collected to enable the detection of weak signals buried in the data. As a consequence, modern radio interferometers produce ever increasing amount of data. A case in point is the Square Kilometre Array (SKA; Dewdney et al. 2009) that is poised to surpass most existing radio interferometers in terms of data output. Interferometric data in raw form are affected by systematic errors such as the receiver beam and the Earth's atmosphere. These errors are determined and corrected during calibration. Calibration of a typical SKA type data set is a heavily compute intensive task because the systematic errors vary with time, frequency and direction (position in the sky). The accuracy of calibration is paramount in the recovery of faint signals of scientific interest. In order to improve the accuracy, we can exploit the continuity of systematic errors across frequency. With the use of consensus optimization (Boyd et al. 2011; Yatawatta 2015; Brossard et al. 2016), it has been shown that calibration can be improved and results based on real observations (Patil et al. 2017) have already confirmed this. The systematic errors are modelled as polynomials in frequency and this model is used as a regularization term in calibration. In order to get the best benefit of consensus optimization, we need to simultaneously calibrate all the data available at all frequencies. In a situation where the number of available compute agents is much less than the number of frequencies at which data are available, this may become problematic. Similar problems have been encountered in radio interferometric imaging where the simultaneous use of all available data is a daunting task for which multiple solutions have been proposed (Deguignet et al. 2016; Meillier, Bianchi & Hachem 2016; Onose et al. 2016; Onose, Dabbech & Wiaux 2017).

We consider a situation where we have P data sets distributed over C compute agents as in Fig. 1. Each data set will have data at one or more contiguous frequencies and we uniquely identify each data set by its central frequency. We assume that each compute agent can process only a single data set at a time, due to resource limitations (e.g. RAM,[1] CPU,[2] GPU[3]). Because of this assumption,

★ E-mail: yatawatta@astron.nl

[1] Random access memory.
[2] Central processing unit.
[3] Graphics processing unit.

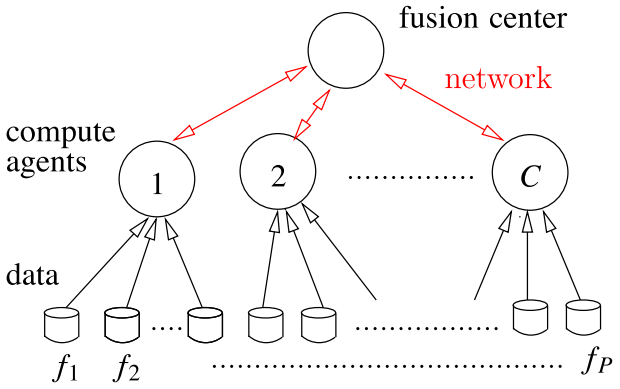

Figure 1. Data are distributed across C compute agents that are connected to the fusion centre via a network. The total number of data sets (P) is larger than the number of available compute agents (C). Each data set will have one or more frequency channels and the central frequencies $f_1, f_2, \ldots, f_P$ uniquely identify each data set.

in our previous work (Yatawatta 2015), we restricted the number of data sets simultaneously processed to match the number of available compute agents. When $P \gg C$, this implies only processing C data sets together (which we call as *a comb*) to reach consensus and repeating this until we have processed all P data sets. However, using all P data sets together to reach consensus is better than using multiple combs of size C. One way to process all P data sets together, when $P \gg C$, is *sequential* processing, where each compute agent uses its resources to sequentially process the data, and consensus is reached using all P data sets. Sequential processing will be slower than processing a single comb, but will give better results. In this paper, we try to achieve the improved result of sequential processing of all P data sets, but with only spending the computational time required to process a single comb of C data sets. In order to do this, we propose a data multiplexing scheme.

Noting that calibration is a non-convex optimization problem, we follow Hong, Luo & Razaviyayn (2015, 2016), where the authors propose a *flexible* alternating direction method of multipliers (ADMM) algorithm. Only a subset of the available data is chosen for processing at each ADMM iteration, which is done in a cyclic manner (each slave cycles through the data one-by-one, also see Section 3.4). The crucial point in multiplexing is the selection of the penalty parameter, and as proposed by Hong et al. (2015, 2016), the penalty needs to be as large as possible. However, unlike in other applications, consensus is achieved based on a model that does not represent the systematic errors with full accuracy. For instance, a simple phase screen will not represent ionospheric errors with sufficient accuracy beyond a certain threshold (Martin, Bray & Scaife 2016). The addition of beam errors (Mort et al. 2017) confound this and systematic errors across a wide field of view cannot be guaranteed to be accurately represented by the consensus polynomials chosen in calibration. Therefore, selecting a penalty parameter that is too high will also give poor results due to the incomplete description of the systematic errors. Hence, we use an adaptive strategy to select the penalty parameter at each ADMM iteration. We base this on the Barzilai–Borwein method (Barzilai & Borwein 1988) as proposed by Xu et al. (2016a,b, 2017). The cyclic selection of different frequencies as in Hong et al. (2016) and the adaptive update of the penalty parameter of each selected data set as in Xu et al. (2016a) are combined in this paper. The initialization of the penalty is done using the Hessian of the cost function as proposed by Yatawatta (2016). We use the minimum description length (MDL; Barron, Rissanen & Yu 2006) as a criterion for selecting the consensus polynomials.

The rest of the paper is organized as follows. In Section 2, we give an overview of direction-dependent calibration using consensus optimization. In Section 3, we introduce the data multiplexing scheme, starting with criteria for selecting the consensus polynomials (Section 3.1), initialization of the penalty parameter (Section 3.2), and adaptive update of the penalty (Section 3.3). In Section 4, we give results based on simulations of an SKA-like telescope (with 512 stations) to demonstrate the performance of the proposed multiplexing scheme.

Notation: upper case bold letters refer to matrices (e.g. $\mathbf{C}$). Unless otherwise stated, all parameters are complex numbers. The set of complex numbers is given as $\mathbb{C}$ and the set of real numbers as $\mathbb{R}$. The matrix pseudo-inverse, transpose, and Hermitian transpose are referred to as $(.)^\dagger$, $(.)^T$ and $(.)^H$, respectively. The identity matrix (of size $N \times N$) is given by $\mathbf{I}_N$. The Frobenius norm is given by $\|.\|$ and the cardinality of a set $\mathcal{F}$ is given by $|\mathcal{F}|$.

2 DIRECTION-DEPENDENT RADIO INTERFEROMETRIC CALIBRATION

We give a concise overview of direction-dependent calibration with consensus optimization in this section. A more comprehensive overview is given in our previous work (Yatawatta 2015; Yatawatta 2016).

2.1 Data model

We consider a radio interferometric array with N dual-polarized receivers. The sky is composed of many discrete sources and we consider calibration along K directions in the sky. The observed data at a baseline formed by two receivers, p and q ($\in [1, N]$), at frequency f is given by Hamaker, Bregman & Sault (1996)

$$\mathbf{V}(pqf) = \sum_{k=1}^{K} \mathbf{J}_{pkf} \mathbf{C}_{pqkf} \mathbf{J}_{qkf}^H + \mathbf{N}_{pqf}, \tag{1}$$

where $\mathbf{V}(pqf)$ ($\in \mathbb{C}^{2\times 2}$) is the observed *visibility* matrix (or the cross-correlations) at frequency f. The systematic errors that need to be calibrated for stations p and q are given by the Jones matrices $\mathbf{J}_{pkf}$, $\mathbf{J}_{qkf}$ ($\in \mathbb{C}^{2\times 2}$), respectively. As calibration is performed along K directions, each station has K Jones matrices associated with it (KN in total for the full array). The uncorrupted sky signal (or *coherency*) along the k-th direction is given by $\mathbf{C}_{pqkf}$ ($\in \mathbb{C}^{2\times 2}$) and is known a priori (Thompson, Moran & Swenson 2001). The values of $\mathbf{J}_{pkf}$, $\mathbf{J}_{qkf}$ and $\mathbf{C}_{pqkf}$ in (1) are implicitly dependent on sampling time and frequency of the observation. However, their variation with f is generally assumed to be smooth and enables the use of consensus optimization. The noise matrix $\mathbf{N}_{pqf}$ ($\in \mathbb{C}^{2\times 2}$) is assumed to have complex, zero mean, circular Gaussian elements.

We use the space alternating generalized expectation maximization (SAGE) algorithm (Fessler & Hero 1994; Kazemi et al. 2011) to estimate $\mathbf{J}_{pkf}$ for all possible values of p and k in (1). This reduces the computational cost and also improves the accuracy (Kazemi et al. 2011). Calibration along the k-th direction is done by using the effective observed data along the k-th direction

$$\mathbf{V}_{pqf}^k = \mathbf{V}(pqf) - \sum_{l=1, l\neq k}^{K} \widehat{\mathbf{J}}_{plf} \mathbf{C}_{pqlf} \widehat{\mathbf{J}}_{qlf}^H, \tag{2}$$

which is calculated using current estimates $\widehat{\mathbf{J}}_{plf}$ and $\widehat{\mathbf{J}}_{qlf}$. This is in fact the *expectation* step of the SAGE algorithm. The *maximization* step of the SAGE algorithm involves minimizing the objective function for the k-th direction defined under a Gaussian noise model as

$$g_{kf}(\mathbf{J}_{1kf}, \mathbf{J}_{2kf}, \ldots) = \sum_{p,q} \|\mathbf{V}^k_{pqf} - \mathbf{J}_{pkf}\mathbf{C}_{pqkf}\mathbf{J}^H_{qkf}\|^2. \quad (3)$$

The summation in (3) is over all the baselines pq that are included in a finite time and frequency interval within which the systematic errors are estimated. It is also possible to modify the objective function for non-Gaussian noise models as in Kazemi & Yatawatta (2013), Ollier et al. (2017) and Grobler et al. (2014).

By using the SAGE algorithm, we are able to separate calibration along K directions to K-independent calibration problems. Therefore, for the sake of simplicity, we drop k from here onwards and rewrite the objective function (3) for any general direction as

$$g_f(\mathbf{J}_f) = \sum_{p,q} \|\mathbf{V}_{pqf} - \mathbf{A}_p\mathbf{J}_f\mathbf{C}_{pqf}(\mathbf{A}_q\mathbf{J}_f)^H\|^2, \quad (4)$$

where $\mathbf{J}_f$ ($\in \mathbb{C}^{2N\times 2}$) is the augmented matrix of Jones matrices for all stations along the k-th direction

$$\mathbf{J}_f \triangleq \left[\mathbf{J}^T_{1kf}, \mathbf{J}^T_{2kf}, \ldots, \mathbf{J}^T_{Nkf}\right]^T, \quad (5)$$

and $\mathbf{A}_p$ ($\in \mathbb{R}^{2\times 2N}$) (and $\mathbf{A}_q$ likewise) is the canonical selection matrix

$$\mathbf{A}_p \triangleq [\mathbf{0}, \mathbf{0}, \ldots, \mathbf{I}_2, \ldots, \mathbf{0}]. \quad (6)$$

Note that only the p-th block of (6) is an identity matrix. To sum up, in direction-dependent calibration along K directions using data at frequency f, we solve K subproblems of minimizing (4).

2.2 Consensus optimization

Consider P data sets distributed across a network of C compute agents as in Fig. 1. Rather than calibrating each data set individually, we use consensus optimization to exploit the continuity of $\mathbf{J}_f$ with f to get an improved result. As introduced in Yatawatta (2015) and Yatawatta (2016), we use ADMM algorithm for consensus optimization. We construct the augmented Lagrangian at the n-th ADMM iteration as

$$L_f(\mathbf{J}^n_f, \mathbf{Z}^n, \mathbf{Y}^n_f, \rho^n_f) = g_f(\mathbf{J}^n_f) + \|\left(\mathbf{Y}^n_f\right)^H(\mathbf{J}^n_f - \mathbf{B}_f\mathbf{Z}^n)\| + \frac{\rho^n_f}{2}\|\mathbf{J}^n_f - \mathbf{B}_f\mathbf{Z}^n\|^2, \quad (7)$$

where we have used the superscript $(\cdot)^n$ to indicate the ADMM iteration number ($n = 1, 2, \ldots$) and the subscript $(\cdot)_f$ denotes data (and parameters) at frequency f. The original cost function $g_f(\mathbf{J}^n_f)$ is given by (4). The Lagrange multiplier is given by $\mathbf{Y}^n_f$ ($\in \mathbb{C}^{2N\times 2}$). The continuity in frequency is enforced by the frequency model given by $\mathbf{B}_f$ ($\in \mathbb{R}^{2N\times 2NF}$), which is essentially a set of basis functions in frequency, evaluated at f. The number of terms used in each basis function is given by F. The global variable $\mathbf{Z}^n$ ($\in \mathbb{C}^{2NF\times 2}$) is shared by data at all P frequencies. The essential difference in (7) from our previous work is that we have the penalty ρ^n_f to be variable both with n and f.

An ADMM iteration for solving (7) is composed of three steps:

$$(\mathbf{J}_f)^{n+1} = \arg\min_{\mathbf{J}} L_f(\mathbf{J}, (\mathbf{Z})^n, (\mathbf{Y}_f)^n, \rho^n_f) \quad (8)$$

$$(\mathbf{Z})^{n+1} = \arg\min_{\mathbf{Z}} \sum_f L_f((\mathbf{J}_f)^{n+1}, \mathbf{Z}, (\mathbf{Y}_f)^n, \rho^n_f) \quad (9)$$

$$(\mathbf{Y}_f)^{n+1} = (\mathbf{Y}_f)^n + \rho^n_f\left((\mathbf{J}_f)^{n+1} - \mathbf{B}_f(\mathbf{Z})^{n+1}\right) \quad (10)$$

that are executed in sequence. Summation across all frequencies at which data are available is denoted by $\sum_f$. The steps (8) and (10) are done for each f in parallel. The update of the global variable (9) is done at the fusion centre. More details of these steps can be found in Yatawatta (2015). In addition to the steps (8), (9) and (10), we also extend (Yatawatta, Diblen & Spreeuw 2017) to update ρ^n_f, which is described in Section 3.3.

The ADMM iterations (8, 9, 10) are initialized as follows.

(i) Normally $\mathbf{J}^1_f$ is initialized using solutions obtained for the previous time interval, or using blocks of identity matrices $\mathbf{I}_2$.
(ii) The Lagrange multiplier $\mathbf{Y}^1_f$ is set to $\mathbf{0}$.
(iii) Since $\mathbf{Z}$ is estimated in closed form, no initialization is necessary.
(iv) We will discuss the initialization of the penalty ρ^1_f in Section 3.2.

Even though all C compute agents calibrate data in parallel, we assume that due to compute resource limitations, only one problem of type (8) can be solved at any time. We consider $P \gg C$ and introduce a multiplexing scheme in Section 3.4 to handle this situation.

3 DATA MULTIPLEXING

We assume that the P data sets are (approximately) evenly divided among the C compute agents, in no particular order. The key point of the data multiplexing scheme is to achieve consensus (9) using all P data sets, regardless of the value of C. We describe various aspects of the proposed multiplexing scheme in the following text.

3.1 Selection of the consensus polynomial model

The consensus polynomial functions used to construct $\mathbf{B}_f$ in (7) are determined in advance. Given a choice of different polynomials, in particular with a varying number of terms F, we can use the MDL (Barron et al. 2006) to select the best polynomial model to use. Let one possible polynomial configuration (with $\widetilde{F}$ number of terms) construct $\widetilde{\mathbf{B}}_f = \widetilde{\mathbf{b}}^T_f \otimes \mathbf{I}_{2N}$ ($\in \mathbb{R}^{2N\times 2\widetilde{F}N}$) at frequency f using $\widetilde{\mathbf{b}}_f \in \mathbb{R}^{\widetilde{F}\times 1}$ that represent the values of the $\widetilde{F}$ basis functions evaluated at f. Let the current solution be $\widetilde{\mathbf{J}}_f$, which can be the solution after the first ADMM iteration (which is essentially the solution without consensus). We calculate the residual sum of squares (RSS) for this solution as

$$\mathrm{RSS} = \frac{1}{8N}\sum_f \rho_f\|\widetilde{\mathbf{J}}_f - \widetilde{\mathbf{B}}_f\mathbf{Z}\|^2, \quad (11)$$

where we have calculated the RSS per parameter (because $\widetilde{\mathbf{J}}_f \in \mathbb{C}^{2N\times 2}$ thus includes $8N$ real parameters). Using the RSS, we find the MDL as

$$\mathrm{MDL} = \frac{P}{2}\log\left(\frac{\mathrm{RSS}}{P}\right) + \frac{\tilde{F}}{2}\log(P) \quad (12)$$

and select the consensus polynomials (in particular $\widetilde{F}$) that give the minimum of (12).

3.2 Initialization of the penalty parameter

The original cost function (4) is non-convex and therefore its Hessian matrix is not positive definite. However, by carefully selecting the penalty parameter ρ_f, we might be able to make the Hessian matrix of the augmented Lagrangian (8) positive definite. Furthermore, the convergence of ADMM also depends on ρ_f (Hong et al. 2015, 2016). The construction of the full Hessian matrix is computationally prohibitive. Therefore, we use the Hessian operator of the cost function (4), which is given by Yatawatta (2015) and Yatawatta (2013a) as

$$\begin{aligned}\mathrm{Hess}_f\left(g_f(\mathbf{J}),\mathbf{J},\eta\right)&=\sum_{p,q}\left(\mathbf{A}_p^T\left((\mathbf{V}_{pqf}-\mathbf{A}_p\mathbf{J}\mathbf{C}_{pqf}\mathbf{J}^H\mathbf{A}_q^T\right)\mathbf{A}_q\eta\right.\\&-\left.\mathbf{A}_p(\mathbf{J}\mathbf{C}_{pqf}\eta^H+\eta\mathbf{C}_{pqf}\mathbf{J}^H)\mathbf{A}_q^T\mathbf{A}_q\mathbf{J}\right)\mathbf{C}_{pqf}^H\\&+\mathbf{A}_q^T\left((\mathbf{V}_{pqf}-\mathbf{A}_p\mathbf{J}\mathbf{C}_{pqf}\mathbf{J}^H\mathbf{A}_q^T)^H\mathbf{A}_p\eta\right.\\&-\left.\mathbf{A}_q(\mathbf{J}\mathbf{C}_{pqf}\eta^H+\eta\mathbf{C}_{pqf}\mathbf{J}^H)^H\mathbf{A}_p^T\mathbf{A}_p\mathbf{J}\right)\mathbf{C}_{pqf}\Big),\end{aligned}\tag{13}$$

where $\eta\in\mathbb{C}^{2N\times 2}$, $\mathrm{Hess}_f\left(g_f(\mathbf{J}),\mathbf{J},\eta\right)\in\mathbb{C}^{2N\times 2}$. Note that η is a matrix that spans the tangent space of the manifold [on which the minima of $g_f(\mathbf{J})$ lie] at $\mathbf{J}$.

To investigate the positive definiteness of the Hessian, we need to find the smallest eigenvalue of (13). Since we have a non-convex cost function, the smallest eigenvalue is negative. As there is no closed form solution for the smallest eigenvalue, we use an iterative approach. First, we define a cost function $h(\eta)$ as (Yatawatta 2016)

$$\begin{aligned}h(\eta)\stackrel{\triangle}{=}\frac{1}{2}&\mathrm{trace}\left(\eta^H\mathrm{Hess}_f\left(g_f(\mathbf{J}),\mathbf{J},\eta\right)\right.\\&+\left.\mathrm{Hess}_f^H\left(g_f(\mathbf{J}),\mathbf{J},\eta\right)\eta\right)\end{aligned}\tag{14}$$

and we find the smallest eigenvalue λ by solving

$$\begin{aligned}&\lambda=\stackrel{\min}{\eta}\ h(\eta)\\&\text{subject to }\ \eta^H\eta=\mathbf{I}_2.\end{aligned}\tag{15}$$

There are several ways to solve (15). In our case, noting that the constraint $\eta^H\eta=\mathbf{I}_2$ makes the minimization of (14) restricted on to a complex Stiefel manifold (Absil, Mahony & Sepulchre 2008), we use the Riemannian trust region method (Absil, Baker & Gallivan 2007; Boumal et al. 2014). The gradient and Hessian of $h(\eta)$ are required to perform this optimization and are given as

$$\mathrm{grad}\left(h(\eta),\eta\right)=\mathrm{Hess}_f\left(g_f(\mathbf{J}),\mathbf{J},\eta\right)\tag{16}$$

and

$$\mathrm{Hess}\left(h(\eta),\eta,\zeta\right)=\mathrm{Hess}_f\left(g_f(\mathbf{J}),\mathbf{J},\zeta\right),\tag{17}$$

where ζ ($\in\mathbb{C}^{2N\times 2}$) is a matrix that spans the tangent space at η of the Stiefel manifold.

Note that the calibration solutions, i.e. $\mathbf{J}$ are kept constant in (14). We use the estimated solutions with ρ_f set to zero and set this as $\mathbf{J}$ in (14). Once we have found the smallest eigenvalue, i.e. λ, we use this as a guideline to select ρ_f. The Hessian of the augmented Lagrangian (7) is given as

$$\mathrm{Hess}_f\left(L_f(\mathbf{J},\mathbf{Z},\mathbf{Y}_f,\rho_f),\mathbf{J},\eta\right)=\mathrm{Hess}_f\left(g_f(\mathbf{J}),\mathbf{J},\eta\right)+\frac{\rho_f}{2}\eta,\tag{18}$$

where $\eta\in\mathbb{C}^{2N\times 2}$ has a similar definition as in (13). So the Hessian of (8) has the smallest eigenvalue $\lambda+\rho_f/2$ where λ is the smallest eigenvalue of (13). By increasing $\rho_f>2|\lambda|$, we can make the minimization (8) convex. However, this also means that the penalty is given more precedence than the actual cost function (4). If the consensus polynomial chosen does not represent the systematic errors entirely accurately, increasing ρ_f larger than $2|\lambda|$ will lead to degraded performance as shown by Yatawatta (2016). Therefore, we initialize ρ_f to a fraction of $|\lambda|$, e.g. $\rho_f^1=|\lambda|/10$ and use $|\lambda|$ as an upper limit for ρ_f in the adaptive update of ρ_f as described in Section 3.3.

The aforementioned initialization of ρ_f is described for one direction out of K directions. Although it is possible to solve (15) for each direction individually, it is much easier to scale the initial value of ρ_f obtained for one direction to match other directions. Consider a rescaling of the model flux in (4), i.e. $\mathbf{C}_{pqf}$ is rescaled to $\kappa\mathbf{C}_{pqf}$, where κ is a positive scale factor. In this case, the solutions $\mathbf{J}_f$ in (4) are scaled to $\frac{1}{\sqrt{\kappa}}\mathbf{J}_f$. Consequently, the penalty term $\frac{\rho_f^n}{2}\|\mathbf{J}^n{}_f-\mathbf{B}_f\mathbf{Z}^n\|^2$ in (7) is also scaled by $\frac{1}{\kappa}$. Therefore, to get back the same penalty, we need to rescale ρ_f to $\kappa\rho_f$. In other words, the penalty is scaled linearly with the scaling of sky model flux. Now consider rescaling of data $\mathbf{V}_{pqf}$ in (4) to $\omega\mathbf{V}_{pqf}$, where ω is a positive scale factor. In this case, the solutions $\mathbf{J}_f$ in (4) are scaled to $\sqrt{\omega}\mathbf{J}_f$ and all terms (both the cost function and the penalty term) in (7) are scaled by ω. Therefore, in this case, no adjustment of ρ_f is necessary.

In summary, the initialization of ρ_f for one selected direction (out of K) is done by first determining the smallest eigenvalue of the Hessian, and using the magnitude of this as the upper bound for ρ_f. For the other $K-1$ directions, the initial values of ρ_f are chosen by linear scaling according to the sky model flux. To minimize the extra compute overhead, the determination of ρ_f need only be performed once for many calibration runs and, where appropriate, can be rescaled to obtain penalty factors for other scenarios.

3.3 Adaptively updating the penalty parameter

In order to increase the convergence rate of ADMM, we update the penalty parameter at each ADMM iteration. Recent work by Xu, Figueiredo & Goldstein (2016b) and Xu et al. (2017) has shown that by using the Barzilai–Borwein (Barzilai & Borwein 1988) method, ADMM can be accelerated in most practical applications. In particular, for non-convex cost functions, Xu et al. (2016a) have shown that this adaptive update gives better results. Motivated by this, we use the same strategy in calibration. We have also compared another popular method for the update of ρ_f, which is called residual balancing (He, Yang & Wang 2000). However, we have found that (Yatawatta et al. 2017) residual balancing is not stable in our case (also found in recent work by Wohlberg 2017). We update the penalty only if we are confident of the performance improvement of ADMM with the update. One way of controlling the update is to use the $|\lambda|$ obtained in (15) as an upper bound for the updated value of ρ_f. We refer to this upper bound as $\overline{\rho}$ in the following text.

The update of ρ_f at the n-th ADMM iteration is done according to algorithm 1. Prior to this update, (8) should be done at each slave and (9) should be done at the fusion centre. The update of ρ_f is done after (10) at each slave. Additional variables used at each slave are $\widehat{\mathbf{Y}}_f^0,\widehat{\mathbf{Y}}_f,\widehat{\mathbf{J}}_f^0\in\mathbb{C}^{2N\times 2}$.

Algorithm 1 Spectral penalty update at the n-th ADMM iteration for data at frequency f

Require: Steps (8), (9) and (10) have been performed to obtain $(\mathbf{J}_f)^{n+1}$. Local variables $\hat{\mathbf{Y}}_f^0, \hat{\mathbf{Y}}_f, \hat{\mathbf{J}}_f^0 \in \mathbb{C}^{2N\times 2}$ are needed for each f (and evolve with ADMM iterations). Upper bound for penalty is $\overline{\rho}$ and $\underline{\alpha} \in (0, 1]$ is a threshold parameter.

1: **if** $n = 1$ **then**
2: Initialize $\hat{\mathbf{Y}}_f^0 := (\mathbf{J}_f)^1$
3: **end if**
4: **if** n is an iteration where ρ_f is updated **then**
5: $(\hat{\mathbf{Y}}_f)^{n+1} := (\mathbf{Y}_f)^n + \rho_f^n \left((\mathbf{J}_f)^{n+1} - \mathbf{B}_f(\mathbf{Z})^n\right)$
6: $\Delta\mathbf{Y}_f := (\hat{\mathbf{Y}}_f)^{n+1} - \hat{\mathbf{Y}}_f^0$, $\Delta\mathbf{J}_f := (\mathbf{J}_f)^{n+1} - \hat{\mathbf{J}}_f^0$
7: $\delta_{11} := \mathrm{trace}\left(\mathrm{Real}\left(\Delta\mathbf{Y}_f^H \Delta\mathbf{Y}_f\right)\right)$
8: $\delta_{12} := \mathrm{trace}\left(\mathrm{Real}\left(\Delta\mathbf{Y}_f^H \Delta\mathbf{J}_f\right)\right)$
9: $\delta_{22} := \mathrm{trace}\left(\mathrm{Real}\left(\Delta\mathbf{J}_f^H \Delta\mathbf{J}_f\right)\right)$
10: $\alpha := \frac{\delta_{12}}{\sqrt{\delta_{11}\delta_{22}}}$, $\alpha_{SD} := \frac{\delta_{11}}{\delta_{12}}$, and $\alpha_{MG} := \frac{\delta_{12}}{\delta_{22}}$
11: $\hat{\alpha} := \begin{cases} \alpha_{MG} & \text{if } 2\alpha_{MG} > \alpha_{SD} \\ \alpha_{SD} - \frac{\alpha_{MG}}{2} & \text{otherwise} \end{cases}$
12: $\rho_f^{n+1} := \begin{cases} \hat{\alpha} & \text{if } \hat{\alpha} \le \overline{\rho} \text{ and } \alpha \ge \underline{\alpha} \\ \rho_f^n & \text{otherwise} \end{cases}$
13: $\hat{\mathbf{Y}}_f^0 := (\hat{\mathbf{Y}}_f)^{n+1}$ and $\hat{\mathbf{J}}_f^0 := (\mathbf{J}_f)^{n+1}$
14: **end if**

Some remarks about algorithm 1 are as follows.

(i) Local variables used (that do not live through ADMM iterations) are $\Delta\mathbf{Y}_f, \Delta\mathbf{J}_f \in \mathbb{C}^{2N\times 2}$, $\delta_{11}, \delta_{12}, \delta_{22}, \alpha, \hat{\alpha}, \alpha_{SD}, \alpha_{MG} \in \mathbb{R}$. The subscripts *SD* and *MG* denote *steepest descent* and *minimum gradient*, respectively (Zhou, Gao & Dai 2006).

(ii) Line 4: the penalty update is not performed at each ADMM iteration, Xu et al. (2017) suggest updating at T periodic values of n, where $T \ge 2$.

(iii) Line 5: this update is different from (10) because $(\mathbf{Z})^n$ is used in the former and $(\mathbf{Z})^{n+1}$ is used in the latter. Therefore, the fusion centre needs to temporarily store the old value of $\mathbf{Z}$ at each iteration.

(iv) Line 12: the threshold $\underline{\alpha} \in (0, 1]$ is used to ensure that the changes in the Lagrange multiplier $\Delta\mathbf{Y}_f$ and solutions $\Delta\mathbf{J}_f$ on line 6 are sufficiently correlated (or have a positive direction cosine). We use $\underline{\alpha} = 0.2$ as in Xu et al. (2017) and by increasing this value, we can restrict the chances of spurious updates.

(v) Line 13: $\hat{\mathbf{Y}}_f^0$ and $\hat{\mathbf{J}}_f^0$ are updated for use during the next update of ρ_f.

The additional computational cost needed to perform the adaptive update of ρ_f is mostly due to three linear operations (lines 5 and 6) and three inner products (lines 7, 8 and 9), all involving matrices in $\mathbb{C}^{2N\times 2}$. In addition, there is increased network communication overhead because the updated values of ρ_f have to be passed to the fusion centre and also because of the additional update on line 5 where $\mathbf{B}_f(\mathbf{Z})^n$ has to be sent from the fusion centre to each slave.

3.4 Cyclic ADMM with data multiplexing

We first introduce the concept of a cyclic buffer. Let $\mathcal{F}$ contain a set of real numbers (e.g. $\mathcal{F} = \{f_1, f_2, f_3\}$) that in our case correspond to a set of frequencies. Consider a function First($\mathcal{F}$): Every time First($\cdot$) is applied on $\mathcal{F}$, it will return the first entry of $\mathcal{F}$ and move this first entry to the last position of $\mathcal{F}$. So if $\mathcal{F} = \{f_1, f_2, f_3\}$, repeatedly calling First($\cdot$) on $\mathcal{F}$ will give us $f_1, f_2, f_3, f_1, f_2, \ldots$, in other words First($\mathcal{F}$) will give us the elements in $\mathcal{F}$ repeatedly, in a cyclic manner.

We use a cyclic buffer to represent the data locally available to each slave, assuming each data set is uniquely identified by its frequency (or central frequency if each data set contains more than one channel). For example, if slave i has data at frequencies $\{f_1, f_2, f_3\}$ locally available, we use $\mathcal{F}_i = \{f_1, f_2, f_3\}$ where $\mathcal{F}_i$ is a cyclic buffer. With the use of a cyclic buffer, we give the pseudo-code for cyclic ADMM in algorithm 2.

Algorithm 2 Cyclic ADMM with data multiplexing

Require: $\mathcal{F}_i$ ($\subset \{f_1, f_2, \ldots, f_P\}$) is a cyclic buffer that represent the data being calibrated by slave i. $\mathcal{W}_i$ is a set of frequencies of the data being calibrated during one ADMM iteration by slave i. M is the maximum number of ADMM iterations. $T (\ge 2)$ is an integer that determines the periodicity of the penalty parameter update.

1: Randomly permute $\mathcal{F}_i$
2: **for** $n = 1, \ldots, M$ **do**
3: **if** $n = 1$ or $n = M$ **then**
4: $\mathcal{W}_i := \mathcal{F}_i$
5: **else**
6: $\mathcal{W}_i :=$ First($\mathcal{F}_i$)
7: **end if**
8: **for** $i = 1, \ldots, C$ in parallel **do**
9: Perform (8) $\forall f \in \mathcal{W}_i$
10: **end for**
11: Perform (9) at the fusion centre
12: **for** $i = 1, \ldots, C$ in parallel **do**
13: Perform (10) $\forall f \in \mathcal{W}_i$
14: **end for**
15: **for** $i = 1, \ldots, C$ in parallel **do**
16: {Decide whether to update the penalty or not}
17: $do\_update := 0$ {Default is no update}
18: **if** $|\mathcal{F}_i| > 1$ **then**
19: $do\_update := 1$
20: **else if** $|\mathcal{F}_i| = 1$ and $n > 1$ and n is a multiple of T **then**
21: $do\_update := 1$
22: **end if**
23: **if** $do\_update = 1$ **then**
24: Perform algorithm 1 to update $\rho_f \; \forall f \in \mathcal{W}_i$
25: **end if**
26: **end for**
27: **end for**

Some remarks on algorithm 2 are as follows.

(i) Line 1: the order of selection of data is randomized for each calibration run. A typical observation will contain many time samples and for each time sample, the ordering of the frequencies of the data calibrated by each slave is random.

(ii) Lines 4, 6: at the first and the last ADMM iterations, (8) and (10) are performed on all available data, possibly in a sequential manner. On the other hand, in all other ADMM iterations, only a single data set is selected for performing these steps. Thus, depending on the ADMM iteration, $\mathcal{W}_i$ can point to all available data sets for slave i or just one data set. Therein lies the multiplexing of data, where in most ADMM iterations, each slave works on a single data set.

(iii) Line 11: step (9) is performed at the fusion centre using all frequencies, regardless of whether the data sets are selected by

the slaves for processing or not. The flexible ADMM algorithm presented in Hong et al. (2016) does not necessarily perform (9) at each ADMM iteration. This in essence converts algorithm 2 to a sequential processing version of ADMM with some delay.

(iv) Line 24: the penalty parameter update is only done on the data at frequencies in $\mathcal{W}_i$, so in most ADMM iterations, for just one data set. Thus, the penalty update interval for any given data set on slave i will be about $|\mathcal{F}_i|$, which is generally larger than the period T.

The performance of algorithm 2 depends on the value of C, especially for solving (9). We investigate this dependence further by using simulations in Section 4.

4 SIMULATIONS

We give results based on simulations of an SKA-like telescope (with $N = 512$ stations) in this section. The configuration of the stations is similar to the one used by Mort et al. (2017) and the integration time is 10 s. We simulate data at $P = 24$ different frequencies, spread in the range 115–185 MHz, but note that in real observations, P could be several hundred or more. The sky consists of $K = 10$ bright point sources, spread across a field of view of 7×7 deg^2, with peak fluxes in the range [1.5, 10] Jy and another 6000 weak sources (which are not known during calibration) with peak fluxes in the range [0.01, 0.1] Jy. Systematic errors along the K directions are randomly generated, with continuity across frequency created by an eight-order ordinary polynomial in frequency and slow variability across time. The 6000 weak sources are clustered (Kazemi, Yatawatta & Zaroubi 2013) around the bright K sources and also corrupted by the systematic errors of each cluster that it belongs to using (1). Finally, complex circular white Gaussian noise is added to the simulated data with a signal-to-noise ratio (ratio of signal power versus noise power) of 10. The consensus polynomial (Bernstein basis functions) has $F = 3$ terms that is chosen according to Section 3.1. Note that this is well below the order of the polynomial that it used to generate the errors.

During calibration, the 6000 weak sources are not used in the sky model and thus they act as additional noise. The systematic errors along the K directions are estimated at each of the P frequencies, per each time sample of 10 s duration. In order to measure the performance of calibration (in particular the convergence of ADMM), we use the error between the ground truth value of $\mathbf{J}_f$ (per direction and frequency) and its estimated value at the n-th ADMM iteration $\widehat{\mathbf{J}}_f^n$, calculated as $\frac{1}{\sqrt{4N}}\|\mathbf{J}_f - \widehat{\mathbf{J}}_f^n\mathbf{U}\|$ with a proper unitary matrix $\mathbf{U}$ ($\in \mathbb{C}^{2\times 2}$) (Yatawatta 2013b), and averaged over all K directions and P frequencies. Additionally, we also calculate the primal residual $\|\mathbf{J}_f^n - \mathbf{B}_f\mathbf{Z}^n\|$ and the dual residual $\|\rho_f^n\mathbf{B}_f(\mathbf{Z}^n - \mathbf{Z}^{n-1})\|$ produced in (8), (9) and (10), which are also averaged over all directions and frequencies. The primal residual represents the error between the systematic errors predicted by the global model and its local estimate at each frequency. The dual residual represents the convergence of the global model to a stable value.

We compare four calibration scenarios using the simulated data. In all cases, the initial values of the penalty parameter is the same and for a source with 1 Jy peak flux, the initial value we use for the penalty is about 10 (determined according to Section 3.2 using data at frequency 148 MHz) and this value is scaled according to the flux for all K sources as described in Section 3.2. The four calibration scenarios are as follows:

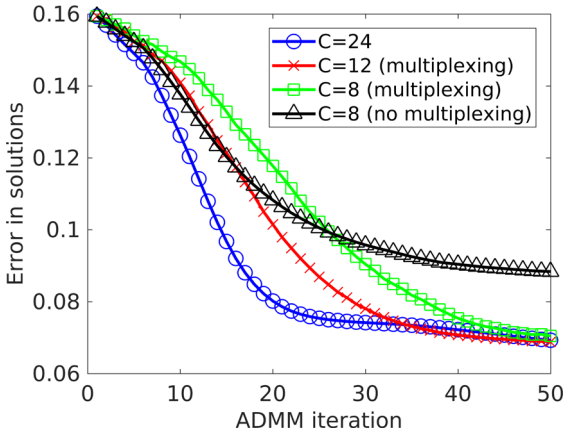

Figure 2. Variation of the error in solutions with ADMM iterations.

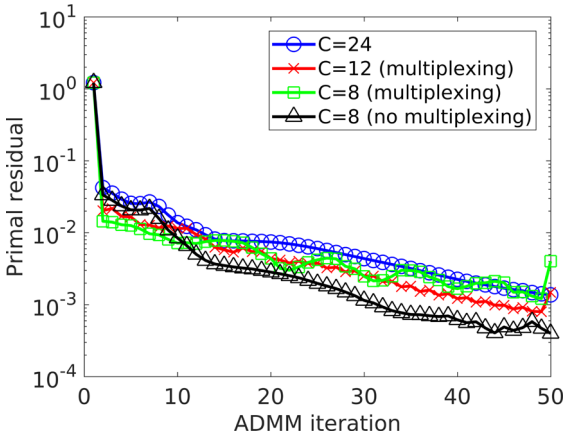

Figure 3. Variation of the primal residual with ADMM iterations.

(i) $C = 24$: calibration using $C = P = 24$ compute agents (no multiplexing but adaptive update of penalty as in algorithm 1 with $T = 2$).

(ii) $C = 12$ (multiplexing): calibration using $C = 12$ compute agents (multiplexing as in algorithm 2).

(iii) $C = 8$ (multiplexing): calibration using $C = 8$ compute agents (multiplexing as in algorithm 2).

(iv) $C = 8$ (no multiplexing): calibration using $C = 8$ compute agents $P/C = 3$ times, (no multiplexing but with adaptive penalty update as in algorithm 1 with $T = 2$) where each comb is made of randomly selected data at C frequencies.

Note that scenario (i) with $C = 24$ is also equivalent to sequential processing of the P frequencies using fewer compute agents if $C < P$. Scenario (iv) is described in Yatawatta (2015, but without adaptive update of the penalty parameter) and this paper aims to improve (Yatawatta 2015), but without the expenditure of additional compute agents nor reverting to sequential processing.

The variation of the error in solutions with ADMM iteration n, compared to the ground truth value is shown in Fig. 2. We see that scenario (i) gives the fastest convergence and the lowest error. Multiplexing with $C = 12$ and $C = 8$ (scenarios ii and iii) gives increasingly slower convergence. On the other hand, scenario (iv) where $C = 8$ and no multiplexing is done gives faster convergence in the beginning, but reaches a higher error floor.

The primal and dual residuals are shown in Figs 3 and 4, respectively. Out of these two, the dual residual variation in Fig. 4 shows

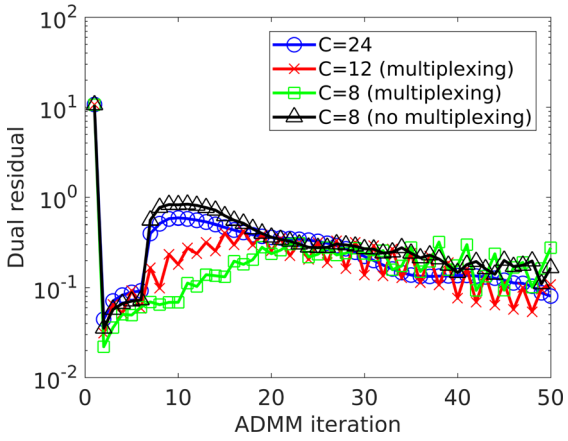

Figure 4. Variation of the dual residual with ADMM iterations.

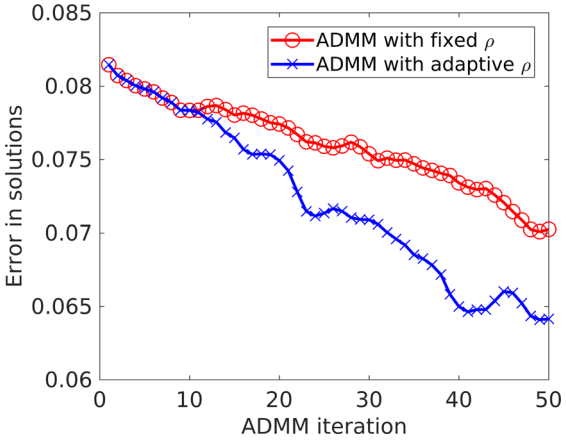

Figure 5. Variation of the error in solutions with ADMM iterations, for fixed penalty parameter and for adaptive penalty parameter.

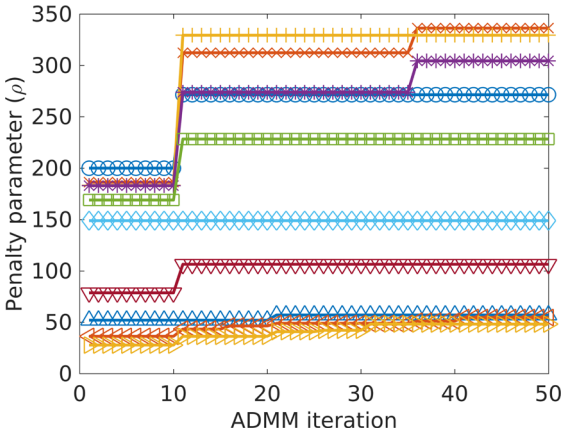

Figure 6. Variation of the penalty parameter ρ with ADMM iterations for all 10 directions at one frequency. The initial values of ρ are scaled according to the flux of the source being calibrated as described in Section 3.2. We see that ρ update occurs at very few instances. Not all directions have updates of ρ and the updates do not happen at the same ADMM iteration for all directions.

clear differences in the four calibration scenarios considered, which can also explain the different convergence rates seen in Fig. 2. First, we see that multiplexing leads to oscillatory behaviour of the dual residual (and also of the primal residual to a lesser extent). This can be explained by the selection of subsets of frequencies for processing as in algorithm 2 at each ADMM iteration, thus leading to a *limit cycle*. Secondly, in Fig. 4, between $n \approx 8$ and $n \approx 20$ we see a marked difference in the dual residual between multiplexing (scenarios ii and iii) and no multiplexing (scenarios i and iv). Higher dual residuals mean faster convergence to the final value of $\mathbf{Z}$ and we see that keeping the frequencies fixed to perform (9) enables faster convergence to the final value of $\mathbf{Z}$. However, because the model $\mathbf{B}_f$ is incomplete, the final value of $\mathbf{Z}$ of each comb (scenario iv) can converge to values different than the final value at convergence using the full data (scenario i). This can also make the primal residual lower for each comb for calibration scenario (iv), which is seen in Fig. 3. This does not mean that the actual error in solutions is lower for scenario (iv), as seen in Fig. 2. At the last ADMM iteration, for scenarios (ii) and (iii), we see an increase in the primal and dual residuals. This is because we use all available data to reach consensus at the last ADMM iteration (see lines 3, 4 and 5 in algorithm 2). However, this does not increase the error in solutions as seen in Fig. 2.

We draw several conclusions from Figs 2–4. First, calibration using all available data (scenario i), either by using more compute agents or by sequential processing, gives the best results. The convergence of data multiplexing (scenarios ii and iii) is slower, mainly because of the convergence of the global variable $\mathbf{Z}$ is slow. However, we can still get the desired accuracy albeit with more ADMM iterations. The main advantage of scenarios (ii) and (iii) as opposed to scenario (i) is the computational cost: either because it uses fewer compute agents or because it requires less computations per each ADMM iteration. To elaborate, consider the total cost required in scenario (i) compared to scenario (iii): in scenario (i), we reach the error floor in about 20 ADMM iteration while in scenario (iii), this is about 40. However, scenario (iii) uses $1/3$ compute agents (or compute cycles in sequential processing). Therefore, the total cost of scenario (iii) is $40/3 \approx 14$, which is less than in scenario (i). Secondly, there is clear improvement in processing all available data (with or without multiplexing as in scenarios i, ii and iii) compared to processing subsets (or combs) of data as in scenario (iv). In other words, it is possible to improve scenario (iv) without expending more computations by multiplexing, but multiplexing will not necessarily give the best achievable result (scenario i). Moreover, the performance of multiplexing degrades as the amount of multiplexing increases, or as fewer compute agents are used to process the same amount of data. This can be seen from comparing the performance of scenario (ii) with scenario (iii).

The effect of the adaptive penalty update is subtle and we emphasize that the penalty is updated only if the new value for the penalty can be estimated with some confidence (that can be controlled by the threshold $\underline{\alpha}$ in 12). This is similar to the conclusions drawn by Xu et al. (2016a). Therefore, in order to compare the effect of the adaptive update of the penalty, we give a comparison of data multiplexing (scenario iii) with and without the adaptive penalty update (also see Yatawatta et al. 2017). We show the error in solutions in Fig. 5, with fixed penalty and with adaptive update of the penalty. Moreover, we show the variation of the penalty parameter at one value of f for this example in Fig. 6. Note that in Fig. 6, the initial value of the penalty is different for each of the K directions (scaled according to the flux) and the variation of the penalty is also different for each direction. None the less, as seen in Fig. 5, the adaptive update of the penalty shows faster reduction in the error in solutions.

Due to the increased number of stations ($N = 512$) and hence the amount of data, an important issue that needs clarification is

the computational cost of calibration. As shown by Kazemi et al. (2011), the scaling of consensus optimization with the number of directions being calibrated K is linear, mainly due to the use of the SAGE algorithm. The scaling with the number of stations N depends on the low-level optimization routine used in consensus optimization. We use the Riemannian trust region algorithm (Absil et al. 2007; Yatawatta 2013a) as the underlying optimization routine. In this algorithm, the linear optimization is done using the truncated conjugate gradient method (Absil et al. 2007) with matrices in $\mathbb{C}^{2N\times 2}$, and therefore the direct solution of a linear system is not needed. This algorithm scales linearly with N because the size of matrices in $\mathbb{C}^{2N\times 2}$ grows linearly with N. The dominating cost is mostly due to the model $\mathbf{C}_{pqf}$ computation in (4) as well as computing the cost function together with its gradient and the Hessian. This needs to be done for each data point and the number of data points scales as N^2 (baselines), and linearly with K and P.

5 CONCLUSIONS

In order to simultaneously process data at a large number of frequencies with a limited number of compute agents, we have proposed a multiplexing scheme for consensus optimization. Based on simulation results, we conclude that the multiplexing scheme together with the adaptive update of the penalty parameter improves the quality of direction-dependent calibration when the compute resources are limited. The source code for the algorithms described in this paper is available at http://sagecal.sf.net/ and https://github.com/nlesc-dirac/sagecal where we have used the message passing interface as our network communication framework. Future work will focus on migrating these algorithms to big-data frameworks such as Apache Spark.

ACKNOWLEDGEMENTS

This work is supported by Netherlands eScience Center (project DIRAC, grant 27016G05) and the European Research Council (project LOFARCORE, grant 339743). We thank the anonymous reviewer and Ronald Nijboer for valuable comments.

REFERENCES

Absil P.-A., Baker C. G., Gallivan K. A., 2007, Found. Comput. Math., 7, 303
Absil P.-A., Mahony R., Sepulchre R., 2008, Optimization Algorithms on Matrix Manifolds. Princeton Univ. Press, Princeton, NJ
Barron A., Rissanen J., Yu B., 2006, IEEE Trans. Inf. Theor., 44, 2743
Barzilai J., Borwein J., 1988, IMA J. Numer. Anal., 8, 141
Boumal N., Mishra B., Absil P.-A., Sepulchre R., 2014, J. Mach. Learn. Res., 15, 1455
Boyd S., Parikh N., Chu E., Peleato B., Eckstein J., 2011, Found. Trends® Mach. Learn., 3, 1
Brossard M., El Korso M. N., Pesavento M., Boyer R., Larzabal P., Wijnholds S. J., 2016, preprint (arXiv:1609.02448)
Deguignet J., Ferrari A., Mary D., Ferrari C., 2016, 24th European Signal Processing Conference (EUSIPCO). EURASIP, p. 1483
Dewdney P. E., Hall P. J., Schilizzi R. T., Lazio T. J. L., 2009, Proc. IEEE, 97, 1482
Fessler J., Hero A., 1994, IEEE Trans. Signal Process., 42, 2664
Grobler T., Nunhokee C., Smirnov O., Van Zyl A., De Bruyn A., 2014, MNRAS, 439, 4030
Hamaker J. P., Bregman J. D., Sault R. J., 1996, A&AS, 117, 96
He B. S., Yang H., Wang S. L., 2000, J. Optim. Theory Appl., 106, 337
Hong M., Luo Z.-Q., Razaviyayn M., 2015, IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, p. 3836
Hong M., Luo Z.-Q., Razaviyayn M., 2016, SIAM J. Optim., 26, 337
Kazemi S., Yatawatta S., 2013, MNRAS, 435, 597
Kazemi S., Yatawatta S., Zaroubi S., Labropoluos P., de Bruyn A., Koopmans L., Noordam J., 2011, MNRAS, 414, 1656
Kazemi S., Yatawatta S., Zaroubi S., 2013, MNRAS, 430, 1457
Martin P. L., Bray J. D., Scaife A. M. M., 2016, MNRAS, 459, 3525
Meillier C., Bianchi P., Hachem W., 2016, 24th European Signal Processing Conference (EUSIPCO). EURASIP, p. 728
Mort B., Dulwich F., Razavi-Ghods N., de Lera Acedo E., Grainge K., 2017, MNRAS, 465, 3680
Ollier V., El Korso M. N., Boyer R., Larzabal P., Pesavento M., 2017, IEEE Trans. Signal Process., 65, 5649
Onose A., Carrillo R. E., McEwen J. D., Wiaux Y., 2016, 24th European Signal Processing Conference (EUSIPCO). EURASIP, p. 1448
Onose A., Dabbech A., Wiaux Y., 2017, MNRAS, 469, 938
Patil A. H. et al., 2017, ApJ, 838, 65
Thompson A., Moran J., Swenson G., 2001, Interferometry and Synthesis in Radio Astronomy, 3rd edn. Wiley, New York
Wohlberg B., 2017, preprint (arXiv:1704.06209)
Xu Z., De S., Figueiredo M., Studer C., Goldstein T., 2016a, preprint (arXiv:1612.03349)
Xu Z., Figueiredo M. A. T., Goldstein T., 2016b, preprint (arXiv:1605.07246)
Xu Z., Taylor G., Li H., Figueiredo M., Yuan X., Goldstein T., 2017, preprint (arXiv:1706.02869)
Yatawatta S., 2013a, IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, p. 3866
Yatawatta S., 2013b, MNRAS, 428, 828
Yatawatta S., 2015, MNRAS, 449, 4506
Yatawatta S., 2016, 24th European Signal Processing Conference (EUSIPCO). EURASIP, p. 265
Yatawatta S., Diblen F., Spreeuw H., 2017, 7th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP) (IEEE CAMSAP 2017). IEEE
Zhou B., Gao L., Dai Y.-H., 2006, Comput. Optim. Appl., 35, 69

This paper has been typeset from a TEX/LATEX file prepared by the author.